9 Views

How to set up CUDA and cuDNN on a dedicated GPU server (Ubuntu 24 guide)

Installation of a server specifically designed to handle deep learning with the help of GPU is supposed to be easy but it seldom is. Problems with driver compatibility, incompatible versions, and incorrect CUDA path are some of the issues that everyone familiar with the process can relate to. The following tutorial will help you to install CUDA and cuDNN with minimum trouble on your Ubuntu 24 OS, irrespective of whether you have a local or remote server via InfinitiveHost or GPU4Host.

Let's get into it.

Why Your Server Location and Provider Matter

Before touching a single terminal command, your hosting choice shapes everything downstream. A dedicated GPU server in Germany behaves differently from one in the UK or Sweden — not just in latency, but in what software comes pre-configured.

The Germany GPU server Ubuntu 24 CUDA cuDNN ready nodes from InfinitiveHost, for example, ship with NVIDIA drivers already validated against the kernel, which saves you a frustrating first hour of troubleshooting. Similarly, the Netherlands GPU server CUDA cuDNN pre-configured node eliminates the manual driver installation step entirely.

If you're working on AI research or production model training, here's a quick rundown of where providers are placing GPU infrastructure right now:

- **UK GPU dedicated server**: cuDNN Ubuntu 24 setups with strong European latency
- **France GPU node**: CUDA toolkit Ubuntu installations with OVH-peered connectivity
- **Sweden GPU server**: cuDNN deep learning environments with green energy infrastructure
- **Switzerland GPU server**: secure CUDA dev environments, strong for privacy-sensitive AI work
- **Ireland GPU server**: Ubuntu CUDA AI workload setups with excellent AWS Transit peering
- **India GPU cloud**: Ubuntu 24 CUDA development servers for the Asia-Pacific market
- **USA GPU server**: CUDA 12 cuDNN Ubuntu 24 setups for North American deployments

Worth mentioning: InfinitiveHost CUDA-ready GPU servers are currently available at 25% OFF, which is a solid entry point if you're provisioning for a long-term project.

Step 1: Verify Your GPU and System Environment

SSH into your dedicated GPU server and start with a clean system audit.

bash

lspci | grep -i nvidia

uname -r

cat /etc/os-release

Type to start searching...

Archive

Select Month

Categories

Select Category

EN

You're looking for Ubuntu 24.04 LTS confirmed, and your NVIDIA GPU visible in the PCI list. If the GPU doesn't show, it's a driver or hardware recognition issue — resolve that before proceeding.

Check whether any NVIDIA drivers are already installed:

bash

nvidia-smi

If this returns a valid output with driver version and GPU details, you're partway there. If it throws an error, you'll install drivers from scratch in the next step.

Step 2: Install NVIDIA Drivers

Update your package index first:

bash

sudo apt update && sudo apt upgrade -y

sudo apt install -y ubuntu-drivers-common

sudo ubuntu-drivers autoinstall

sudo reboot

After reboot, confirm the driver is loaded:

bash

nvidia-smi

You should see your GPU model, driver version, and CUDA version listed. On a properly configured dedicated GPU server, this step is often already done for you — but it's worth verifying.

Step 3: Install the CUDA Toolkit

For Ubuntu 24, CUDA 12.x is the current production-stable release. Head to the NVIDIA CUDA repository and add it manually:

bash

wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2404/x86_64/cuda-keyring_1.1-1_all.deb

sudo dpkg -i cuda-keyring_1.1-1_all.deb

sudo apt update

sudo apt install -y cuda-toolkit-12-4

Once installed, add CUDA to your PATH. Open your .bashrc:

bash

nano ~/.bashrc

Add all these below-mentioned lines at the bottom:

bash

export PATH=/usr/local/cuda/bin:$PATH

export LD_LIBRARY_PATH=/usr/local/cuda/lib64:$LD_LIBRARY_PATH

Source it:

bash

source ~/.bashrc

Check out installation:

bash

nvcc –version

You should see CUDA 12.x confirmed. This is the core of what makes a dedicated GPU server useful for model training — without a working CUDA toolkit, the GPU is just sitting idle.

Step 4: Install cuDNN

cuDNN is what neural network frameworks like TensorFlow and PyTorch actually use for optimized GPU computation. NVIDIA has simplified the installation process considerably for Ubuntu 24.

bash

sudo apt install -y libcudnn9-cuda-12

EN

Or for cuDNN 8 (required by some older TensorFlow versions):

bash

sudo apt install -y libcudnn8 libcudnn8-dev

Verify cuDNN is installed correctly:

bash

dpkg -l | grep cudnn

GPU4Host CUDA environment server configuration tips recommend pinning your cuDNN version to match your framework. TensorFlow 2.13+ works cleanly with cuDNN 8.9 on CUDA 12, while PyTorch 2.x supports cuDNN 9.x natively.

Step 5: Validate the Full Stack

Run a fastest Python validation to validate CUDA and cuDNN are talking to each other in the right way:

bash

python3 -c "import torch; print(torch.cuda.is_available()); print(torch.cuda.get_device_name(0))"

For TensorFlow:

bash

python3 -c "import tensorflow as tf; print(tf.config.list_physical_devices('GPU'))"

Both should return your GPU name and validate CUDA access. If you're on an InfinitiveHost CUDA-ready node or a USA GPU server CUDA 12 cuDNN Ubuntu 24 setup, this validation step typically passes cleanly without additional troubleshooting.

Common Issues Worth Knowing

Driver and CUDA version mismatch is the single most common failure point. Each CUDA version requires a minimum driver version. CUDA 12.4 needs driver 550.54 or higher.

Kernel header mismatch happens after system updates. If nvidia-smi breaks after an apt upgrade, reinstall the DKMS module:

bash

sudo apt install -y dkms nvidia-dkms-550

Wrong Python environment — always activate your virtual environment before testing. System Python and conda environments don't share CUDA paths.

On any well-configured **dedicated GPU server** — whether it's a Switzerland GPU server secure CUDA dev environment or an Ireland GPU server Ubuntu CUDA AI workload setup — these issues are minimized because the base kernel and driver pairing is pre-validated.

Conclusion

Getting CUDA and cuDNN running cleanly on a dedicated GPU server comes down to three things: the right provider, the right version pairing, and the right installation order. Ubuntu 24 is stable and well-supported by NVIDIA's current toolkit, making it the right base OS for serious AI work in 2026.

Whether you're running a France GPU node CUDA toolkit Ubuntu installation, a Sweden GPU server cuDNN deep learning environment, or starting fresh on a USA GPU server CUDA 12 cuDNN Ubuntu 24 setup — this guide gets you from zero to a verified working environment. For managed setups with pre-configured drivers, InfinitiveHost CUDA-ready GPU servers at 25% OFF are worth serious consideration for teams who'd rather spend time training models than debugging kernel modules.

Related Blogs

EN

How to Run Llama 3 / Llama 4...

infi_admin | July 2, 2026

How to Run Llama 3 / Llama 4 on a Dedicated GPU

Multimodal AI on GPU Dedicated Servers (Vision +...

infi_admin | June 30, 2026

Multimodal AI on GPU Dedicated Servers (Vision + Text + Audio) Try

GPU Server vs CPU Server for Deep Learning:...

infi_admin | June 24, 2026

GPU Server vs CPU Server for Deep Learning: When Does GPU Actually

Running LLMs on Dedicated GPU Servers: Llama, Mistral...

infi_admin | June 19, 2026

Running LLMs on Dedicated GPU Servers: Llama, Mistral & Custom AI Deployment

Best GPU Dedicated Server for Minecraft, Rust &...

infi_admin | June 17, 2026

Best GPU Dedicated Server for Minecraft, Rust & Game Server Hosting (2026)

AI-Powered Video Transcoding on GPU Dedicated Servers: A...

infi_admin | June 12, 2026

AI-Powered Video Transcoding on GPU Dedicated Servers: A 2026 Deep Dive Video

Load More

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

Comment *

Save my name, email, and website in this browser for the next time I comment.

Post Comment

EN

PREV
When Your Live Stream Keeps Buffering (And How...

INFINITIVE HOST

Our Address: Office No: 201-202, Second Floor, Elements Mall, DCM Ajmer Road, Jaipur, 302021, Rajasthan, India

US: NO .1910, Thomes Avenue, Cheyenne, Wyoming. 82001

Around the Web

An ISO 27001 & 9001 Certified Company

Platforms

- AWS
- Google Cloud
- Azure
- Kubernetes
- Odoo
- Vtiger Solutions

Company

- About Us
- Our Team
- Work Culture
- Awards & Recognitions
- Certificates
- Payment Options
- FAQs

Digital Solutions

- Web Development
- App Development
- Digital Marketing

Solutions

- Server Management
- Backup Solutions

Email & Security

- Office 365
- Zimbra
- SSL
- Cyber Security

© 2026 Infinitive Host. All rights reserved

EN